# A PDE based approach to multi-domain partitioning and quadrilateral meshing

Nicolas Kowalski[1], Franck Ledoux[1], and Pascal Frey[2]

[1] CEA, DAM, DIF, F-91297 Arpajon, France
kowalskin@ocre.cea.fr, franck.ledoux@cea.fr
[2] UPMC Univ Paris 06, UMR 7598, Laboratoire J.-L. Lions, F-75005, Paris, France frey@ann.jussieu.fr

**Summary.** In this paper, we present an algorithm for partitioning any given 2d domain into regions suitable for quadrilateral meshing. It can deal with multi-domain geometries with ease, and is able to preserve the symmetry of the domain. Moreover, this method keeps the number of singularities at the junctions of the regions to a minimum. Each part of the domain, being four-sided, can then be meshed using a structured method. The partitioning stage is achieved by solving a PDE constrained problem based on the geometric properties of the domain boundaries.

**Key words:** mesh generation; finite element methods; partial differential equations, elliptic; quadrilateral; partitioning; multi-domain

## 1 Introduction

In numerous computational engineering applications, such as automobile crash simulations, structural mechanics, neutronics or fluid-structure interactions, quadrilateral meshes are preferred over triangular meshes [1]. Moreover, some numerical simulations can require several materials, gaz or fluids. The meshes are then referred as being multi domain.

### 1.1 Problem statement

The most desirable properties of quadrilateral meshes usually are:

1. to minimize the number of discrete singularities, defined here as the non 4-valent vertices (internal vertices that do not have exactly 4 neighbors);
2. to align elements along the boundary, i.e. to have several layers of quadrilaterals aligned along the boundary in a parallel fashion (e.g. boundary layers);
3. to achieve high-quality elements, i.e. to have each quadrilateral as close to a square or a rectangle as possible;

4. and to have the elements abide by a prescribed size map.

In case of multi domain simulations, meshes may have to comply wit a fifth desirable property: the mesh may be conform along the interface lines between incident domains. Then the line interface must be explicitly discretized in the mesh [3]. Generating meshes fulfilling all those properties is a difficult task where the optimization of one property may be in contradiction with the optimization of another one. Hence, a compromise between the various constraints and requirements is often required. Several numerical and algorithmic solutions have been proposed over the last decades to generate quadrilateral meshes abiding by those properties. Those methods can be categorized as either structured [2, 3] or unstructured [4, 5, 6, 7, 8, 9], depending on the importance they give to the underlying mesh structure. On the one hand, structured methods are unable to mesh most geometries and to deal with boundary discretization constraints along interface lines. On the other hand, unstructured methods usually yield meshes lacking some structure and having many undesired singularities. A good compromise between the geometrical flexibility of fully unstructured meshes and the numerical efficiency achieved on globally structured meshes can be achieved through block-structured meshes. Such meshes can be considered as unstructured arrangements of quadrilateral regions meshed in a structured way.

Block-structured meshes have one main advantage over other kinds of meshes: they can theoretically adapt to any given domain while maintaining the good element quality associated with structured meshes. Moreover, block-structured meshes have two important added benefits that no unstructured mesh generation method would dispute. First of all, efficient solvers can be employed within individual regions due to their logical rectangular structure [10]. Second of all, block-structured meshes have a hierarchical memory structure, allowing processors to have dedicated memories with quick access containing the whole information they need.

However, to the best of our knowledge, no algorithm has been developed to provide a good enough partitioning of an arbitrary domain into four-sided regions. Most of the attempts to provide such an algorithm have more or less revolved around the use of the medial axis of the domain [11]. However, the medial axis is numerically unstable, in that small changes of the domain geometry can greatly perturb the obtained partitioning. Moreover, the resulting regions are generally not four-sided, and thus are not optimal for quadrilateral meshing. Even for a domain as simple as a square, a medial axis decomposition provides triangular parts, and not the grid one would expect.

The approach we propose here is an attempt to get a block-structured partitioning that overcomes the main drawback of the previous developed techniques.

3. In the case of numerical simulations where contact and sliding effects between materials are modeled it is preferable to have duplicated vertices and edges in the contact areas.

### 1.2 Overview of our approach

Given a 2d domain $\Omega$ of arbitrary shape, we introduce a new automatic domain partitioning method where the geometric features of $\partial\Omega$ (see Figure 1-a) are propagated in $\Omega$ by solving a PDE (see Figure 1-b). By this way, we ensure the three first properties given previously to be satisfied. Our process assumes a uniform triangulation $T_h$ of $\Omega$, that is used to solve a PDE problem that yields a directionality field (see Figure 1-c and Section 2). Singularities of this field are then identified numerically and lines connecting singularities are deduced from the directionality field. Using these lines, our approach achieves a partition of $\Omega$ into four-sided regions (see Figure 1-c and Section 3). This algebraic approach to domain partitioning can be used on arbitrary 2d domains, as well as with domains presenting interfaces or multiple closed curves delimiting boundaries. Moreover it preserves the symmetry of $\Omega$ and minimizes the number of singularities. Once such a partitioning is defined, each region can then be easily meshed through a bilinear transfinite interpolation. The resulting block-structured meshes are usually of very high quality (see Figure 1-d and Section 4). Our method, however, cannot create meshes with a prescribed varying size map. It creates meshes with constant size elements, thus lacking the fourth properties a mesh shall exhibit. One way to overcome this drawback may to partition the domains and then to mesh each quadrilateral region using an unstructured method dealing with varying size maps[12].

## 2 Creation of a directionality field

The main idea of our method consists in using a unit vector field prescribed on $\partial\Omega$ that we propagate on the whole $\Omega$. This propagation is performed by solving a PDE with Dirichlet conditions defined on $\partial\Omega$. Numerically, this PDE is solved by applying a P1 finite element method onto a triangulation $T_h$ of $\Omega$ with the Dirichlet conditions given on the vertices of $\partial T_h$. Note that the interfaces between different domains are considered as being part of the geometry boundary in the following. A first intuition would be to use the outer normal vector field defined on the boundary. However, such a field is discontinuous at a corner with an angle of $\frac{\pi}{2}$, whereas the desired mesh at such a corner is perfectly regular. To overcome this problem, we use the concept of *directionality* [13, 14], which is continuous on such corners and thus better fits the expected structure of quadrilateral meshes.

### 2.1 Definition of the directionality of $\Omega$

A regular vertex $P$ of a quadrilateral mesh of $\Omega$, that does not belong to $\partial\Omega$, is 4-valent, and the 4 edges connected to $P$ can be seen as two sets of opposite edges. Each of these sets, in turn, can be seen as the local discretization of a curve, with $P$ being the intersection of two curves. Our aim being to

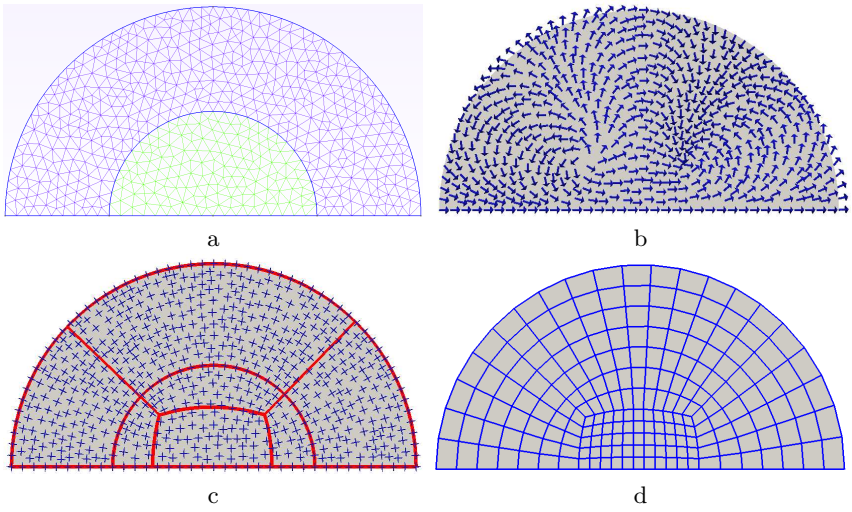



**Fig. 1.** *Overview of the main steps of our algorithm. In a, the background mesh used. The geometry has two domains with distinct mesh colors. In b, the unit representation vector field obtained by solving a PDE. In c, the corresponding cross field and the domain partitioning into four-sided regions. In d, a full quadrilateral mesh is obtained by using a bilinear transfinite interpolation over each region.*

generate quadrilateral elements, we assume that the the two directions of a cross must be orthogonal one to the other. The two tangent directions of these curves at $P$ can thus be described by a *cross*, that we call the *directionality* of the mesh at $P$ (see Figure 2). This notion of cross field is often used in computer visualization nowadays [13], and is usually defined as follows :

$$C_\theta = \left\{ \mathbf{v}_k = \left( cos(\theta + \frac{k\pi}{2}), sin(\theta + \frac{k\pi}{2}) \right)^T, 0 \leq k \leq 3 \right\}, \ with\ \theta \in [0, \frac{\pi}{2}[. \tag{1}$$

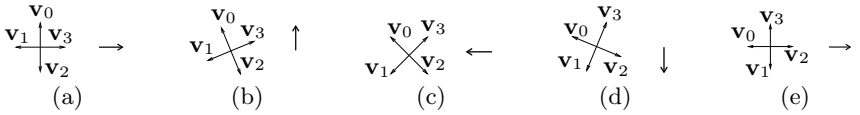



**Fig. 2.** *Examples of crosses along the corresponding representation vectors. Notice that the crosses in (a) and (e) have the same representation vector, and that the representation vector may be different from any vector of the cross.*

An important point to notice is that the cross field is not related to the reference axis used to compute angles. Indeed, choosing as reference axis a

direction that makes an angle $\theta_0$ with the X-axis is equivalent to rotating $\Omega$ by an angle $-\theta_0$. In that case, when each cross rotates by an angle $-\theta_0$, each representation vector rotates by an identical angle of $-4\theta_0$, thus leaving the problem globally unchanged.

### 2.2 Definition of the representation vector field as the solution of a non-linear PDE

Our goal is to propagate an information known on the boundary of the domain inside the domain in a smooth fashion. The solution of this problem can be considered as the steady-state of a conduction problem

$$\frac{\partial \mathbf{u}}{\partial t} - \mathrm{div}(k\nabla \mathbf{u}) = 0 \tag{2}$$

With $k$ constant, we get the Laplace equation. Solving this problem consists here in minimizing a functional $J$ defined as:

$$\begin{cases} J(\mathbf{u}) = \int_\Omega |\nabla \mathbf{u}|^2 dx, \\ \mathbf{u}(x) = \mathbf{u}_0(x) \quad \forall x \in \partial\Omega, \end{cases} \tag{3}$$

where $x$ is a vector function and the Dirichlet condition is defined at a point $P$ of $\partial\Omega$ as the representation vector corresponding to the cross defined by tangents and normals to $\partial\Omega$ at $P$. This guarantees that the cross field is locally "aligned" with each boundary in its vicinity. To initialize this vector at every $C^0$ corner $C$ of $\partial\Omega$, we arbitrary define it at $C$ as the average of the representation vectors of the two geometric edges incident to $C$.

Solving this problem is easy using FEM or other numerical methods. It is an elliptic problem, well-posed with good properties as ensuring the maximum principle on $\mathbf{u}$. However, in order to keep vectors unitary in the context of this work, we add a non-linear constraint. Indeed, allowing norms to vary introduce a bias leading to non-constant variations of the crosses orientations, which is not desirable as only directions are meaningful in our problem. Thus, given a geometrical domain $\Omega$, we minimize the functional $J$ defined as [4]:

$$\begin{cases} J(\mathbf{u}) = \int_\Omega |\nabla \mathbf{u}|^2 dx, \\ \mathbf{u}(x) = \mathbf{u}_0(x) \quad \forall x \in \partial\Omega, \\ |\mathbf{u}(x)| = 1 \quad \forall x \in \Omega. \end{cases} \tag{4}$$

As such, unlike Problem 2, Problem 4 is ill-posed, as a solution may not exists. Indeed, the solution of the same problem without considering the non-linear constraint may result in some cases in a vector field having one or more zeroes; but as the non-linear constraint prevents the vector field from having any zero, the resulting vector field is not defined at some points. Instead of solving Problem (4), we find an approximate solution defined on $\Omega$ deprived of a finite number of points by proceeding in two steps:

4. here, the used norm is the Froebenius norm.

– first, we solve problem (4) while neglecting the non-linear constraint;
– then, this approximation is refined by linearizing of the norm constraint.

### 2.3 Solving the linear approximation of the problem

By neglecting the non-linear constraint in the third equation, Problem (4) becomes a simple steady-state heat problem. Classically, we introduce a weak formulation of this problem:

$$\forall \mathbf{v} \in V,\ \int_{\Omega} \nabla \tilde{\mathbf{u}} \nabla \mathbf{v}\ dx = -\int_{\Omega} \nabla \mathbf{u}_0 \nabla \mathbf{v}\ dx \tag{5}$$

with $V = H_0^1(\Omega)$ being the space of $H^1$ functions vanishing on $\partial\Omega$, $\tilde{\mathbf{u}} \in V$ and $\mathbf{u} = \tilde{\mathbf{u}} + \mathbf{u}_0$. The Lax-Milgram theorem guarantees the existence and the uniqueness of a solution to this problem.

Numerically, we solve this problem using a Galerkin finite element approach on a triangulation $T_h$ of $\Omega$ with $P_1$-Lagrange elements. This yields a linear problem $A \times x = b$. Since matrix $A$ is symmetric and positive-definite, the vector $x$ is computed using a conjugate gradient method [15].

### 2.4 Linearization of the norm constraint

The non-linear constraint $|u| = 1$ is then considered. Due to the non-linearity of this constraint, functional $J$ is no longer convex and its argmin is zero, and, as it has a lower bound of zero, it means that it has several local energy minima. In our approach, we find the local minima closest to our first approximated solution using a linear approximation of the norm constraint [16] based on Lagrange multipliers. The algorithm is based onto the fact that normalizing vectors with a norm greater than one reduces the value of the functional $J$. In practice, we use an iterative process starting with the solution of Problem 3 as initial solution. Each step of this process is divided in two parts, a PDE resolution and a normalization process, each of them decreasing the value of $J$. In the following, the term $\mathbf{v}^n(p)$ denotes the value of the vector $\mathbf{v}$ at point $p$ during the step $n$. Considering $\mathbf{v}$ as a solution to Problem 5, we have $|\mathbf{v}^n(p)| = 1$ at each point $p$ of the triangulation $T_h$. In order to compute the solution at step $n+1$, we add a linear constraint to each point $p \in T_h$:

$$\mathbf{v}^{n+1}(p) \cdot \mathbf{v}^n(p) = 1. \tag{6}$$

and, using the conjugate residuals method, we solve the previous steady-state heat problem 3 under constraints given by Equation 6 at each point $p \in T_h$. We introduce Lagrange multipliers [17] to add the linear constraint at each point leading to the following problem:

$$\begin{pmatrix} A & C \\ C & 0 \end{pmatrix} \begin{pmatrix} u \\ \lambda \end{pmatrix} = \begin{pmatrix} b \\ 1 \end{pmatrix}, \tag{7}$$

with 1 the column matrix having all its elements equal to 1, $\lambda$ the column matrix, which coefficients $\lambda_p$, $p \in T_h$, corresponds to the Lagrange multiplier associated to the constraint $|\mathbf{v}(p)| = 1$, and $C$ a diagonal matrix, such that $C(2p, 2p) = \alpha$ and $C(2p+1, 2p+1) = \beta$, for $\mathbf{v}^n(p) = (\alpha, \beta)$. Adding those constraints changes the initial problem into an optimization problem under constraint. Eventually, we normalize all obtained vectors, thus decreasing the value of the functional $J$ as the constraint ensures us that $|\mathbf{v}^{n+1}(p)| \geq 1$.

*Remark 1.* Except the traditional numerical errors that are due to the discretization, the triangulation $T_h$ that we use to solve the Problem 4 has no significative influence on the structure of the generated directionality field with regard to the convergence and the stability of the procedure.

## 3 Domains partitioning

At completion of the first step, we obtained a piecewise affine, continuous unit vector field defined at the vertices of $T_h$. It can be extended by interpolation to define a continuous vector field over $\Omega$. This vector field satisfies a prescribed Dirichlet boundary condition on $\partial T_h$. Each of these representation vectors corresponds to a cross, leading to a smooth cross field $CF$ defined over all $T_h$. The general idea of our approach consists in analyzing the topology of this cross field. This means identifying its singularities, or zeroes, and connecting them, thus emphasizing its underlying structure. We will also consider the geometric singularities of $T_h$, which are the $C^0$ corners of the domains, so as to obtain a partition of $T_h$ into quadrilateral regions.

The notion of topological skeleton of a vector field is well-known [18], and can be understood as a set of curves representing the defining features of the vector flow, for example, the set of curves integrated from saddles points along the directions of the eigenvectors. We extend it here to cross fields, in a similar manner to what has been suggested in other works [13]. A semantic difference is that we use this topological structure to partition the geometric domain.

### 3.1 Singularities of a directionality field

**Definition 1.** Given a vector field $\mathbf{v}$ defined on $\Omega$, a point $x \in \Omega$ such that $\mathbf{v}(x) = 0$ is a singularity.

By extension, in our context, the singularities of the directionality field are detected as the zeroes of the piecewise linear interpolation of the representation vector field defined at the vertices of $T_h$. Moreover, we include the discontinuities of the domain boundaries as singularities. Considering these intrinsic geometric singularities allows us to obtain a fully quadrilateral partitioning of the domain (see Section 3.3). We recall here some definitions and propositions [19] related to the Poincaré index of the zeroes of a vector field.

**Definition 2.** The Poincaré index of an isolated zero $x$ of a 2d vector field $\mathbf{v} = (v_1, v_2)$ is the number of rotations of the vector field while traveling in positive direction along any closed curve enclosing $x$ but no other singularity. Put another way, for $\gamma$ a closed curve containing $x$ and no other singularity, the Poincaré index $i_x$ of $P_0$ is defined as:

$$i_x = \frac{1}{2\pi} \oint_\gamma d\phi, \ with \ \phi = arctan \frac{v_1}{v_2}. \tag{8}$$

**Proposition 1.** Let $\mathbf{v}$ be a vector field on a 2d geometrical domain $\Omega$. Suppose $\mathbf{v}$ is defined as a linear interpolation of non-zero values given at the vertices of a triangulation, $T_h$ of $\Omega$, all singularities are necessarily of first order, i.e. their index is either 1 or $-1$.

**Proof:** The result is exhibited by enumerating the different possibilities and summing the angular rotations of the field along the edges of the triangle around the singularity.

### 3.2 Streamlines and separatrices

A streamline of a vector field is a special kind of field line that can be defined as follows:

**Definition 3.** Let $\mathbf{v}$ be a $C^1$-continuous vector field defined over a domain $\Omega$, and let $\gamma(s), \ s \in [0,1]$ be the parametrization of a $C^1$ curve defined over $\Omega$, with $\partial\gamma(s)$ its derivative. Then $\gamma$ is a streamline of $\mathbf{v}$ if

$$\forall \ s \in [0,1], \partial\gamma(s) \times \mathbf{v}(\gamma(s)) = 0, \tag{9}$$

where "$\times$" denotes here the cross product of two vectors. A streamline is thus a $C^1$-continuous curve for which the tangent to the curve at any given point has the same direction as the vector field value at this point. It is to be noted that there is exactly one streamline going through any regular point of a vector field, and that if $\gamma$ is not a closed curve then both end points of $\gamma$ are either a singularity of the representation vector field or lie on $\partial\Omega$.

Now we extend this concept to cross fields. Around any regular point of a cross field, this field can locally be described as 4 vector fields. At singular points, such a separation of the components of the crosses is impossible. Thus, in regular areas (containing no singularity), locally to a small neighborhood of each point, it can be described as the orthogonal intersection of two streamlines. Those streamlines can be computed, starting from this point, by selecting one of the 4 vectors defined at this point and, doing so, by selecting one of the 4 smooth vector fields and then using the same numerical scheme one would use to compute a vector streamline. By repeating this process, one can compute whole streamlines on the field as long as they do not go in the vicinity of field singularities. We can now give a suitable definition for streamlines in cross fields.

**Definition 4.** Let $CF = (\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4)$ be a continuous cross field defined over a domain $\Omega$, except for a finite number of singularities[5]. Let $\gamma(s)$, $s \in [0,1]$ be a parametrization of a $C^1$ curve defined over $\Omega$, with $\partial\gamma(s)$ its derivative. Then $\gamma$ is a streamline of $CF$ if

$$\forall\ s \in [0,1], \exists i \in [1..4], \partial\gamma(s) \times \mathbf{v}_i(\gamma(s)) = 0. \tag{10}$$

For each singularity $S$, we build the separatrices, which are the streamlines of the cross field ending on $S$ (the red streamlines in Figure 3-a, for example).

**Definition 5.** A streamline $\gamma$ is a separatrix if there is a singularity $S_0$ such that $\exists\ t \in [0,1], \gamma(t) = S_0$. In that case, we say that $\gamma$ is a separatrix of $S_0$.

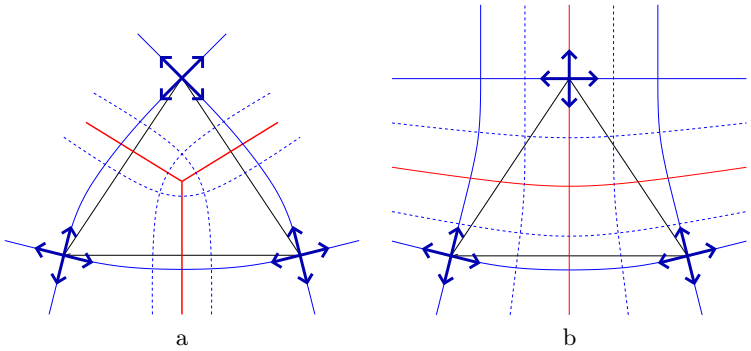



**Fig. 3.** *In a and b, streamlines in a triangle containing a singularity or not.*

To compute the separatrices at a singularity $S_0$, we have to find a streamline $\gamma$ such that $\gamma(t) = S_0$ for $t \in [0,1]$. Numerically, in the triangulation $T_h$ of $\Omega$, we apply the following process. As a singularity $S_0$ is necessarily located in a triangle $T_0$ of $T_h$ or on an edge separating two triangles $T_1$ and $T_2$ of $T_h$, any separatrix of $S_0$ crosses the edges of $T_0$ in the first case and the edges of $T_1$ and $T_2$ in the latter case. Let us consider the first case, the latter case being a straightforward extension. Considering a singularity $S_0$ located into a triangle $T_0 \in T_h$ defined by its 3 vertices $S_1$, $S_2$ and $S_3$. We traverse all the edges of $T_0$ to find the points where separatrices of $S_0$ intersect the edges of $T_0$. Let $P$ be a point of the edge $[S1, S2]$. Considering the cross field $CF$, we compute the value $CF(P)$ by linearly interpolating the value of $CF(S_1)$ and $CF(S_2)$. The value of $CF(P)$ corresponds to a cross or to 4 vectors $\mathbf{v}_k$, $0 \leq k \leq 3$ representing the directions of the streamlines at $P$. Let $\mathbf{u}(P)$ be the vector $\mathbf{S_0P}$. Hence, a streamline going through $P$ is a separatrix of $S_0$ *iff*

5. we remind the reader here that this means that the representation vector field of $F$ is $C^1$-continuous, and not that any of the $\mathbf{v}_i$ field is $C^1$-continuous

$$\exists k \in [0..3], \frac{\mathbf{v}_k}{|\mathbf{v}_k|} = \frac{\mathbf{u}(P)}{|\mathbf{u}(P)|}. \tag{11}$$

In this case, the tangent of the streamline at $P$ is $\mathbf{v}_k$ (see Figure 4).

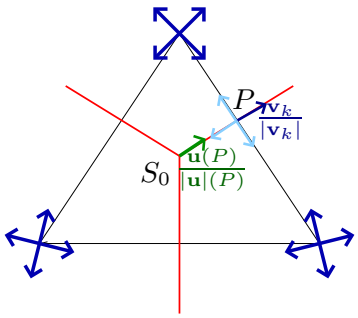



**Fig. 4.** *The intersection of a separatrix (in red) of the singularity $S_0$ with the triangle around $S_0$, at a point $P$.*

Applying this process for every singularities of $CF$, we define all the intersection points between the triangles containing a singularity and the associated separatrices. All these points are on the boundary of the triangulation $T'_h = T_h - \{T_0, \ldots, T_n\}$ with $\{T_0, \ldots, T_n\}$ the sets of triangles containing the singularities of $CF$. By construction, $CF$ is regular on $T'_h$. Then the expected separatrices can be approximated on $T'_h$ using the following Runge-Kutta numerical scheme until either the boundary of the geometry or another triangle containing a singularity is met. Considering a triangle $T \in T'_h$, a point $X_i \in \partial T$ and a direction $\mathbf{d}_i$ at $X_i$, our aim is to find the intersection point $X_{i+1}$ between $\partial T$ and the line defined by the direction $\mathbf{d}_i$ and the point $X_i$. Hence the direction $\mathbf{d}_{i+1}$ in $X_{i+1}$ should be computed (see Figure 5-a and 5-b).

To this end, a representation vector in $X_i$ is linearly interpolated from the representation vectors of the vertices $S_i$ of $T$. The vector $\mathbf{v}_{\mathbf{d}_i}$ of the corresponding cross $C_i$ that is closest to the input direction $\mathbf{d}_i$ is selected. The selection of this vector is propagated onto all crosses interpolated over $T$, thus defining a vector field $Y = f(X)$ over $T$ that is consistent with $\mathbf{d}_i$ (see Figure 5-a). Then we use the Heun's method [20], an easy-to-use variation of a Runge-Kutta method, to integrate the ODE $Y = f(X)$ over $T$, starting from $X_i$. A line can thus be built using segments, by iterating this integration process over the successive encountered triangles. It is to be noted that, even though the singularities of the representation vector field and the ones of the cross field are identical, their separatrices are totally different.

Another interesting point is that the Poincaré index of a singularity $S$ of the representation vector field and the number of separatrices of $S$ are closely related. Each singularity of the vector field having an index of $+1$ corresponds

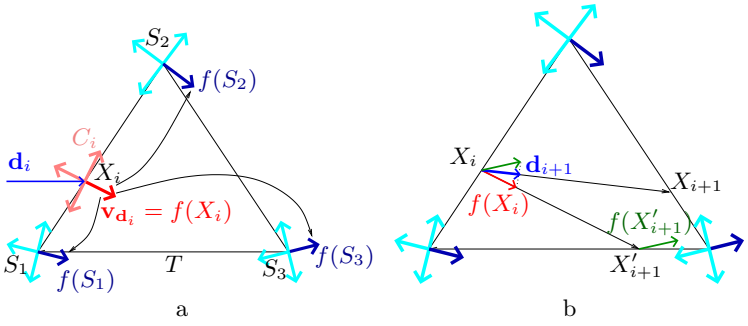



**Fig. 5.** *Illustration of the integration process over a triangle.*

to a singularity of the cross field having 3 separatrices, and each singularity of the vector field having an index of $-1$ corresponds to a singularity of the cross field having 5 separatrices, while regular points of the vector field correspond to points where a cross is defined and therefore are the intersection of two streamlines in the cross field (see Figure 6).

### 3.3 Definition of the domain partitioning.

Considering a domain $\Omega$ and a cross field defined over $\Omega$, the separatrices of this cross field along with their corresponding singularities allow us to partition $\Omega$. Every separatrix ends either on a singularity or on $\partial\Omega$, there is no guarantee that the obtained regions are four-sided. In fact, it only provides regions that can be meshed through a submapping technique [21], i.e. regions without any inner singularities. In order to get four-sided regions, we extend the set of of representation vector field singularities by adding a set of geometric singularities, which are defined as the $C^0$ corners on $\partial\Omega$. We then build separatrices for those geometric singularities by using the same process than for the representation vector field singularities.

**Proposition 2.** The set of separatrices of the geometric and field singularities leads to a partition of $\Omega$ into 3 or 4-sided regions. The 3-sided regions necessarily contain a singularity.

**Proof:** By construction, all the regions we obtained contains no singularity except eventually on their corners. This means that a parametrization $f(u, v)$ of the inside of a region is always possible. Moreover, the boundaries of the considered region can always be included inside the parametrization by continuity, except if one edge of the region is of null length, which corresponds to a triangular region. In this case, one of the two sets of parallel streamlines of the region has all its streamlines that converge to a same point, which is the degenerated edge of the region.

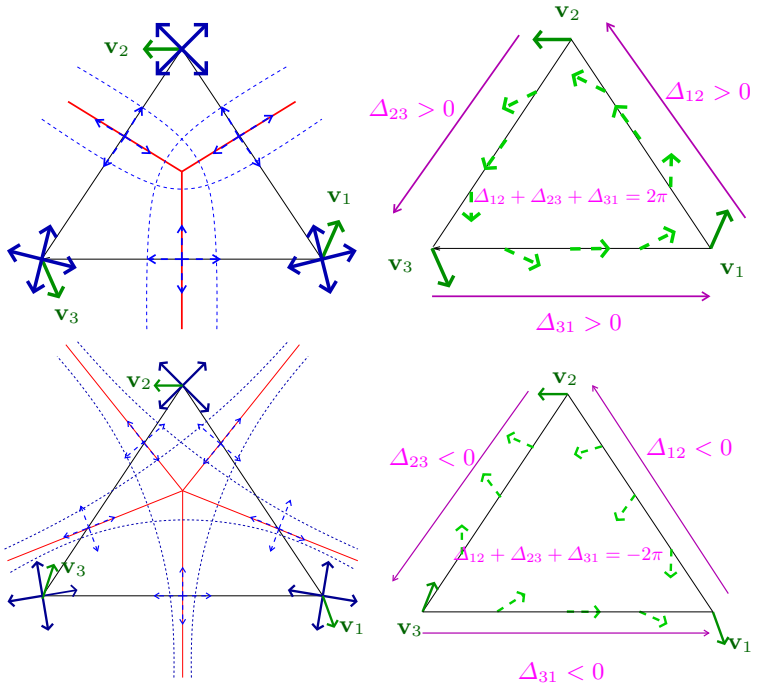



**Fig. 6.** *A 3-valent singularity, on top, and a 5-valent singularity, on bottom. In blue, the local cross field, and in green the corresponding representation vector field. Separatrices are in red, along with the shape of surrounding streamlines in blue dots. On the right, in light green dots, the linear interpolation of the representation vectors along the edges is shown. The path along which we compute the index is the triangle around the singularity.*

For all the streamlines to behave like this, the angle at this corner of the region must be at maximum $\frac{\pi}{4}$, by continuity of the cross field inside the region. As the directions of the separatrices going from an inner singularity are uniformly distributed, due to the piecewise linear nature of the cross field, and as we only have 3 and 5-valent singularities, the minimum angle for a region corner that is a singularity inside $\Omega$ is $\frac{2\pi}{5}$. It follows that the degenerated edge of the considered region is a geometric corner featuring a small angle. In the absence of such a degenerated edge, the region can be parametrized by a one to one mapping function $f(u, v)$, with $u \in [0, 1]$ and $v \in [0, 1]$.□

Note that, while the computational cost of the method is strictly related to the number of elements in the background triangulation, the time needed to partition the domain for the different examples presented in this paper varied from about one second, for most of them, to twenty seconds for the most complicated ones.

# 4 Results

Once each of the domain has been partitioned into four-sided parts, each of these parts is meshed using a transfinite bilinear interpolation. This meshing process creates a parametrization of the region that coincides with the boundary parametrization. It leads to an immediate grid meshing process by considering any discretization of the parameter spaces of $u$ and $v$. It shall be noted that the computational cost of this step, even for millions of elements, is negligible with respect to the cost of the domains partitioning step.

## 4.1 Quality comparison

While it is not easy to define a quality measurement for a domain partitioning, it is a lot more convenient to define one for quadrilateral meshes. In this section, we compare meshes obtained with our approach with meshes obtained using an unstructured algorithm provided by the software GMSH [22]. Both the minimum angle measure and the scaled Jacobian measure are used to compare the results. Both meshes have about 9000 quadrilaterals. On Figure 7, we present the minimum angle measure as color maps for each mesh. On Figure 8, we do the same for the scaled Jacobian measure and, on Figure 9, we presents histograms of those results. For both measures, we get better results with -our approach. It is mainly due to the weak number of singularities that arises in the partitioning.

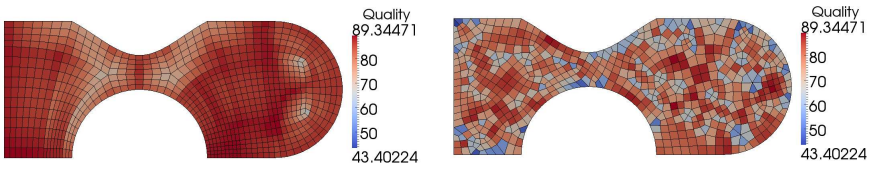
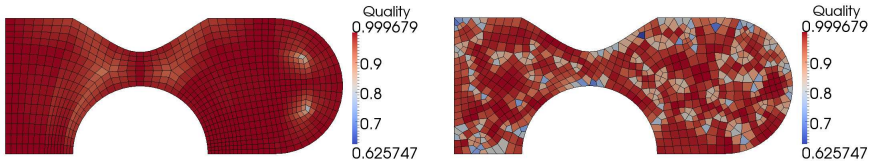



**Fig. 7.** *Color map of the minimum angle quality measure on our mesh (in a) and on the unstructured mesh (in b).*

## 4.2 Single domain examples

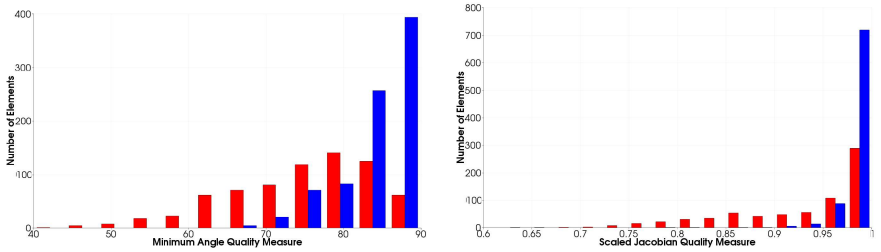
The first example of Figure 10 illustrate the symmetry preservation of our approach. We can also note the weak number of singularities: four 3-valent singularities and three 4-valent ones.

In Figure 11, we show how the domain partitioning evolves when the geometry of the domain is slightly modified. In this case, the domain in a rectangular area where two quarter of circles have been removed. In Fig. 11-a,




a b

**Fig. 8.** *Color map of the Jacobian quality measure on our mesh (in a) and on the unstructured mesh (in b) as a color map.*




a b

**Fig. 9.** *Histograms representing the repartition of elements by quality, the unstructured mesh being represented in red and our mesh in blue. In (a) the histogram represents the repartition for the minimum angle quality, and in (b) it represents the repartition for the Jacobian quality.*

the two quarters of circles are symmetric and then we obtain a symmetric quadrilateral mesh. In Fig. 11-b, the vertical side of the right quarter of circle is greater and the two singularities that are inside the domain are not more connected. In Fig. 11-c, the difference between the two quarters of circle is more important but the global topology of the domain partitioning is similar.

The last example of Figures 12 and 13 is a two-dimensional aerofoil section consisting in three elements, a leading-edge flat, a main element and a trailing-edge flap. In order to compute flows around these three elements, it is better to get boundary layers of quadrilateral cells along the the aerofoil. It is what we get in Fig. 12-b where just few singularities appear around the aerofoil. In Figure 13, we highlight two different regions of the final mesh. The boundary layers we generate allows to get cells orthogonal to the boundary.

### 4.3 A multi-domain example

In Figure 14, two incident domains are partitioned separately. Even if the obtained meshes have a similar element size, the common curve is not discretized in the same way (see Fig. 14-b). We can also see that the partitioning lines of each domain do not cross the common curve (see Fig. 14-a).

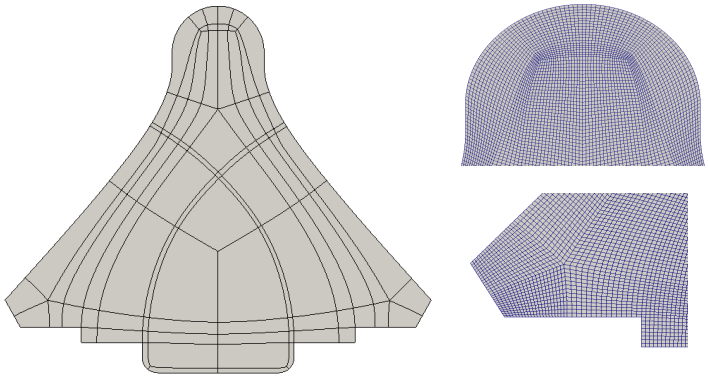


**Fig. 10.** *A symmetric domain partitioning on the left and two focuses on the obtained quadrilateral mesh*

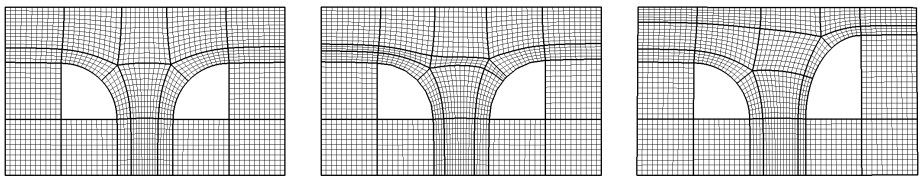



**Fig. 11.** *Additional examples.*

On the contrary, in Figure 15, the two domains were partitioned and discretized together. Thus, we obtain a contact line that was discretized only once and a conformal mesh all over both domains. It may be noted that unlike in Figure 14, the separatrices emanating from a singularity of a domain can spread through the other domain.

## 5 Conclusion

In this paper, we presented a novel approach to domain partitioning, which can be used as an efficient block-structured quadrilateral mesh generation algorithm. This fully automatic algorithm uses the solution of a PDE to extend the information contained by normals and tangents at the boundary inside the domain. It generates a partition of the domain into four-sided regions with curvilinear edges that can be easily meshed into quadrilateral elements using a transfinite bilinear interpolation. The number of singular vertices in the resulting mesh is minimal or close to minimal. Being an algebraic method at its core, our method provides results that are independent of the initial

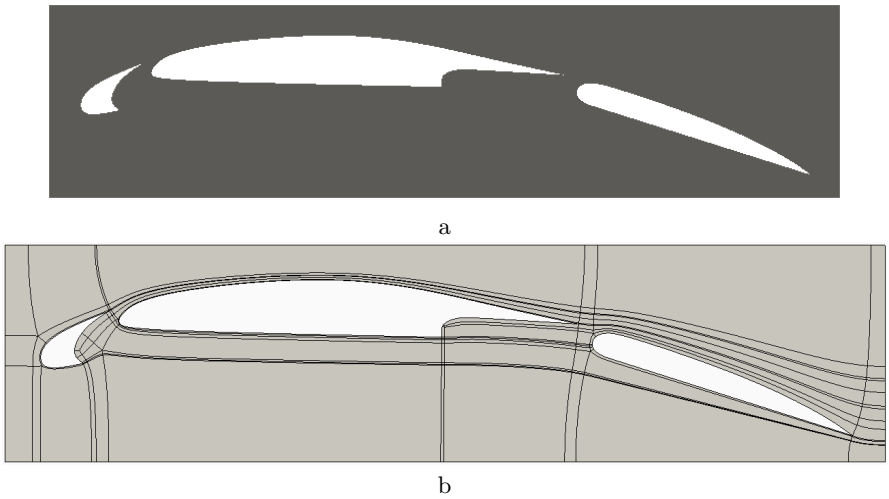


**Fig. 12.** *A 2d space around an aerofoil in a and and the corresponding domain partitioning in b*

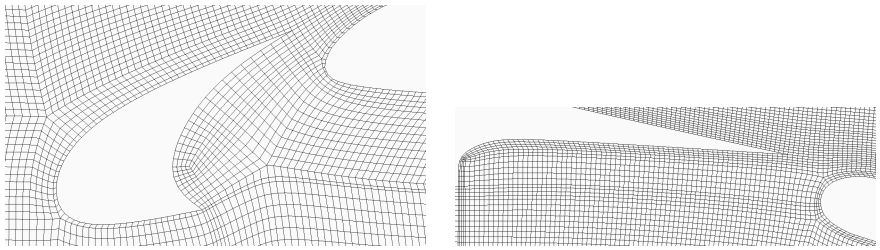


**Fig. 13.** *Mesh generated in area of interest for the space around an aerofoil*

orientation of the domain and that respects its geometrical symmetries. An added benefit of our method is that the main computational cost is due to the domain partitioning. As a result, the additional cost for having a more refined mesh is very low, making our method quicker than most other methods for refined meshes. Should the need arise, both the partitioning and the meshing processes have data structures and implementation allowing for parallelization. Although the meshing part of the process is currently unable to deal with varying prescribed element sizes, future work is scheduled to improve this by using known methods to refine the mesh inside desired regions.

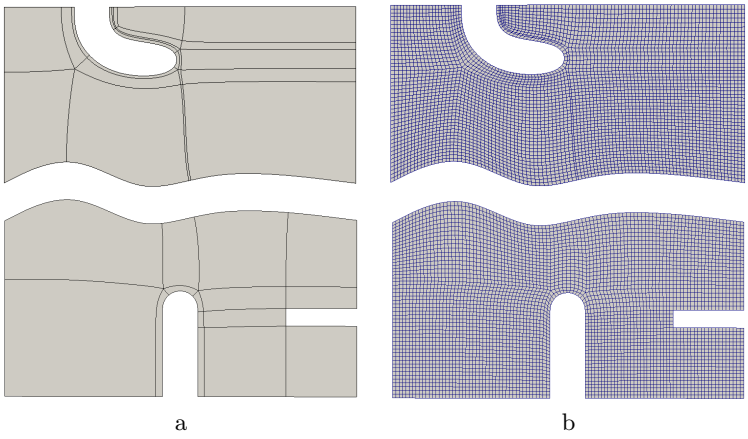



**Fig. 14.** *Two domains sharing a curve are discretized separately: domain partitioning in (a) and quadrilateral meshes in (b).*

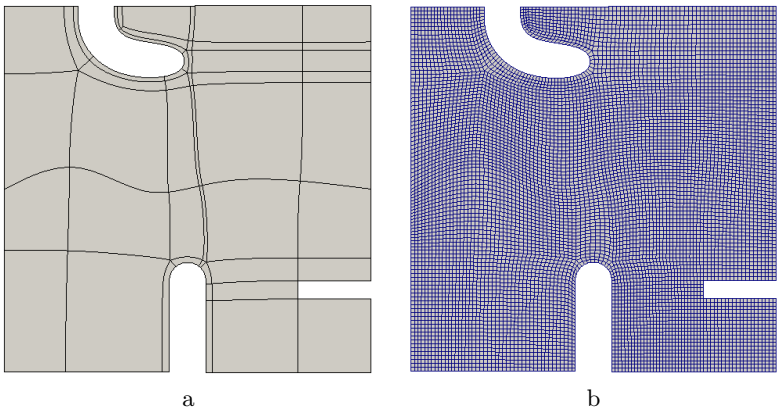



**Fig. 15.** *The domains given in Fig. 14 are jointly meshed: domain partitioning in (a) and quadrilateral mesh in (b).*

## References

1. Cook RD, Malkus DS, Plesha ME. *Concepts and Applications of Finite Element Analysis, 3rd edition.* Wiley, New York, 1989.
2. Faux ID, Pratt MJ. *Computational geometry for design and manufacture.* Ellis Horwood, Chichester, 1979.
3. Whiteley M, White D, Benzley S, Blacker T. Two and Three-Quarter Dimensional Meshing Facilitators. *Engineering With Computers* 1994; **12**:144–145.
4. Schneiders R. A grid-based algorithm for the generation of hexahedral element meshes. *Engineering with Computers* 1996; **12**:168–177.

5. Staten ML, Owen SJ, Blacker T. Unconstrained Paving and Plastering: A New Idea for All Hexahedral Mesh Generation. *Proceedings of the International Meshing Roundtable* 2005; **14**:399–416.
6. Borouchaki H, Frey PJ. Adaptive triangular-quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering* 1998; **41**:915–934.
7. Owen SJ, Staten ML, Canann SA, Saigal S. Q-morph: An indirect approach to advancing front quad meshing. *International Journal for Numerical Methods in Engineering* 1999; **44**:1317–1340.
8. Lévy B, Liu Y. Lp Centroidal Voronoi Tesselation and its Applications. *ACM Transactions on Graphics* 2010.
9. Kowalski N, Ledoux F, Staten ML, Owen SJ. Fun sheet matching: towards automatic block decomposition for hexahedral meshes. *Eng. with Comp.* 2010.
10. Lindquist D.R., Gilest M.B. A comparison of numerical schemes on triangular and quadrilateral meshes, *11th Int. Conf. on numerical methods in fluid dynamics, Lecture Notes in Physics*, 1989;**323**:369–373.
11. Tam TKH, Armstrong CG. 2D finite element mesh generation by medial axis subdivision. *Adv. in Eng. Software and Workstations* 1991; **13**:313–324.
12. Remacle JF, Lambrechts J, Seny B, Marchandise E, Johnen A, Geuzaine C. Blossom-quad: a non-uniform quadrilateral mesh generator using a minimum cost perfect matching algorithm. *International Journal for Numerical Methods in Engineering* 2012.
13. Ray N, Vallet B, Li W, Lévy B. N-symmetry direction field design. *ACM Transactions on Graphics* 2008.
14. Palacios J, Zhang E. Rotational symmetry field design on surfaces. *ACM Transactions on Graphics* 2007.
15. Saad Y. *Iterative methods for sparse linear systems*. SIAM 2003.
16. Alouges F. A new algorithm for computing liquid crystal stable configurations: the harmonic mapping case. *SIAM Journal on Numerical Analysis* 1997; **34**:1708–1720.
17. Bramble JH. The Lagrange multiplier method for Dirichlet's problem. *Mathematics of Computation* 1981; **37**.
18. Helman JL, Hesselink L. Vizualizing Vector Field Topology in Fluid Flows. *IEEE Computer Graphics and Applications* 1991, **11**:36–46.
19. Tricoche X, Scheuermann G, Hagen H. Continuous Topology Simplification of Planar Vector Fields. *Proceedings of IEEE Visualization* 2001; 159–166.
20. Ascher UM, Petzold LR. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM 1998.
21. White DR. Automated Hexahedral Mesh Generation by Virtual Decomposition. *Proceedings, 4th International Meshing Roundtable* 1995; 165–176.
22. Geuzaine C, Remacle JF. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering* 2009, **79**:1309–1331.